Remote Power Control of Wireless Network Interfaces

Andrea Acquaviva Tajana Simunic† Vinay Deolalikar† Sumit Roy†

DEIS - University of Bologna
Viale Risorgimento 2
Bologna, 40136, Italy

†HP Laboratories
1501 Page Mill Road
Palo Alto, 94304 CA, USA

Abstract— **This paper presents a new power management technique aimed at increasing the energy efficiency of client-server multimedia applications running on wireless portable devices. We focus on reducing the energy consumption of the wireless network interface of the client by allowing the remote server to control the power configuration of the network card depending on the workload. In particular, we exploit server knowledge of the workload to perform an energy-efficient traffic reshaping, without compromising on the quality of service. We tested our methodology on the SmartBadge IV wearable device running an MPEG4 streaming video application. Using our technique we measured energy savings of more than 67% compared to no power management being used on the WLAN interface. In addition, we save as much as 50% of energy with respect to the standard 802.11b power management. All of the energy savings are obtained with no performance loss on the video playback.**

I. Introduction

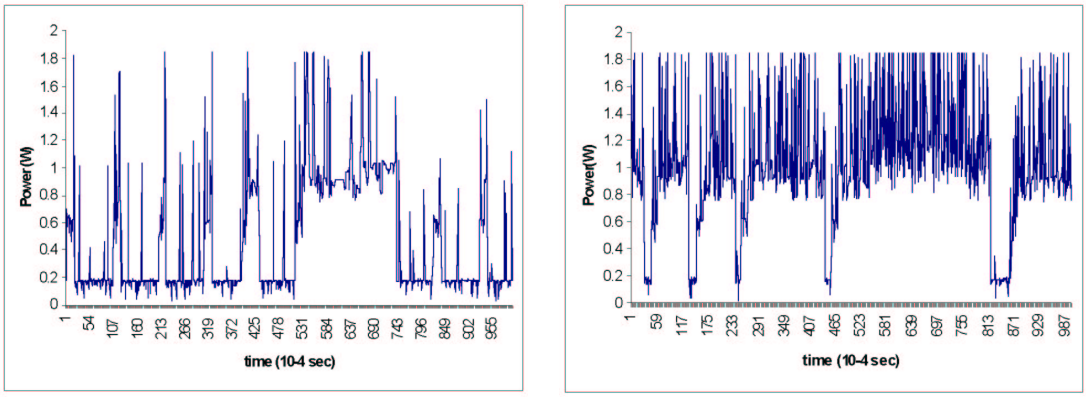
Portable devices spend a considerable amount of energy in order to support power hungry peripherals e.g. wireless local area network (WLAN) interfaces, and liquid crystal displays (LCD). A good example of such a device is the SmartBadge IV [22] wearable computer. It consists of a StrongARM-1110 processor and SA-1111 coprocessor, memory, WLAN interface, audio codec and 2.2" LCD. Depending on the network traffic, the WLAN accounts for as much as 63% of the overall system power consumption.

One way to reduce the energy consumption is to use the power management included in the 802.11b standard (802.11b PM) [5]. In the standard, an access point (AP) transmits a beacon every 100 ms, followed by a traffic indication map (TIM). Each client checks the TIM for its turn to send or receive data. When not communicating, the WLAN goes into the doze mode until the next beacon. Unfortunately, the protocol power management is not very effective. First, the energy efficiency of the 802.11b PM decreases and receiver wait times increase with more mobile hosts, since multiple concurrent attempts at synchronization with the beacon cause media access contention. Second, the response time of the wireless link with 802.11b PM grows because of the delay imposed by sleep periods [23]. These two issues can be resolved by careful scheduling of communication between the server and the client WLAN. Lastly, in a typical wireless network, broadcast traffic can significantly reduce the chances to enter the doze mode. Figure 1 shows the power consumption of a WLAN card with 802.11b PM enabled under light and heavy network broadcast traffic conditions. Clearly, as the amount of broadcast traffic increases, the WLAN spends a large amount of energy listening to it, even if no other application is running on the device. As a result, very little or no energy savings are obtained. One way to solve this problem is to turn off the card. It is important to schedule data transmission carefully, since the overhead of waking up the WLAN from the off state is large.

Many current wireless local area networks are organized in a client-server fashion. Multiple WLAN clients connect to wired servers via APs. Servers are great candidates for efficient scheduling of data transmission to clients as they are not power constrained, and know both wired and wireless network conditions.

In this work, we present a server controlled power management strategy. Our technique exploits server knowledge of the workload, traffic conditions and feedback information from the client in order to minimize WLAN power consumption. Our methodology is applicable to a wide variety of applications, ranging from video and audio streaming, to web browsing and e-mail. We define two new entities: a server power manager (server PM) and a client power manager (client PM). Server PM uses the information obtained from the client and the network to control the parameters of 802.11b PM and to perform energy efficient traffic reshaping so that WLAN can be turned off. Client PM communicates through a dedicated low-bandwidth link with the server PM and implements power controls by interfacing with device drivers. It also provides a set of APIs that client programs can use to provide extra information to the server.

In order to illustrate the effectiveness of our approach, we tested our methodology with a streaming video application. By using our approach with this application, we can exploit the server knowledge of stream characteristics. We show that when our methodology is implemented on both the server end, and on the client end, we measure savings of more than 67% in power with respect to leaving the card always on, and more than 50% relative to using default 802.11b PM. Even larger savings are possible for applications that inherently have longer idle periods, such as e-mail or web browsing. Our methodology can also be



a) light traffic b) heavy traffic

Fig. 1. 802.11 PM under different broadcast traffic conditions

easily extended to manage other system components (e.g. CPU).

The rest of the paper is organized as follows. Section II presents some related work. Section III gives an overview of the proposed methodology and describes server and client power managers. Section V presents experimental results and Section VI concludes the paper.

II. Related Work

The wireless network power optimization problem has been addressed at different abstraction layers, starting from physical, to system and application level. Energy efficient channel coding and traffic shaping to exploit battery lifetime of portable devices were proposed in [4]. A physical layer aware scheduling algorithm aimed at efficient management of sleep modes in sensor network nodes is illustrated in [18]. Energy efficiency can be improved at the data link layer by performing adaptive packet length and error control [9]. At the protocol level, there have been attempts to improve the efficiency of the standard 802.11b, and proposals for new protocols [6], [7], [20]. Packet scheduling strategies can also be used to reduce the energy consumption of transmit power. In [15] authors propose the E^2WFQ scheduling policies based on Dynamic Modulation Scaling. A small price in packet latency is traded for the reduced energy consumption.

Traditional system-level power management techniques are divided into those aimed at shutting down components and policies that dynamically scale down processing voltage and frequency [21], [1]. Energy-performance trade-offs based on application needs have been recently addressed [8]. Several authors exploit the energy-QoS tradeoff [13], [23], [12]. A different approach is to perform transcoding and traffic smoothing at the server side by exploiting estimation of energy budget at the clients [17]. A new communication system, consisting of a server, clients and proxies, that reduces the energy consumption of 802.11b compliant portable devices by exploiting a secondary low-power channel is presented in [19]. Since multimedia applications are often most demanding of system resources, a few researchers studied the cooperation between such applications and the OS to save energy [10], [2], [16], [11].

We present a new methodology, where server knowledge of the workload is exploited to control the power configuration of the radio interface. Compared to physical and protocol layer strategies, the power control is performed at the application level, so it does not require hardware modifications. Compared to client-centric approaches, we exploit additional information available at the server, and thus obtain large energy savings without loosing performance. Moreover, with respect to previous application-driven policies, our infrastructure can be used with a wide range of applications, since it exploits very common parameters.

III. Server Controlled Power Management

Our methodology exploits server knowledge of the workload, traffic conditions and feedback information from the client in order to minimize power consumption. The server schedules communication sessions with the client based on the knowledge of both, the wireless and wired networks, e.g favorable channel conditions or channel bandwidth capabilities. When broadcast traffic needs to be monitored, the server can enable the 802.11 PM. Alternatively, it can coordinate the shut down of WLAN and perform on-time wake-up, thus avoiding the performance penalty typically incurred by client-centric approaches. Our application driven infrastructure can be also be used to manage power consumption in stand-alone and ad-hoc applications. The power control strategy can easily be extended to include other system components, such as peripherals or the CPU.

In order to exploit the extra information available at the server and the client, the traditional client-centric power manager model has to be extended. Two different power managers are defined: one running on the client (Client PM) and the other on the server (Server PM). The two PMs exchange power control information through a dedicated TCP connection. As shown in Figure 2, the client PM interfaces directly with the drivers on the portable device. It also collects client application dependent information. The server PM interfaces with the server application and

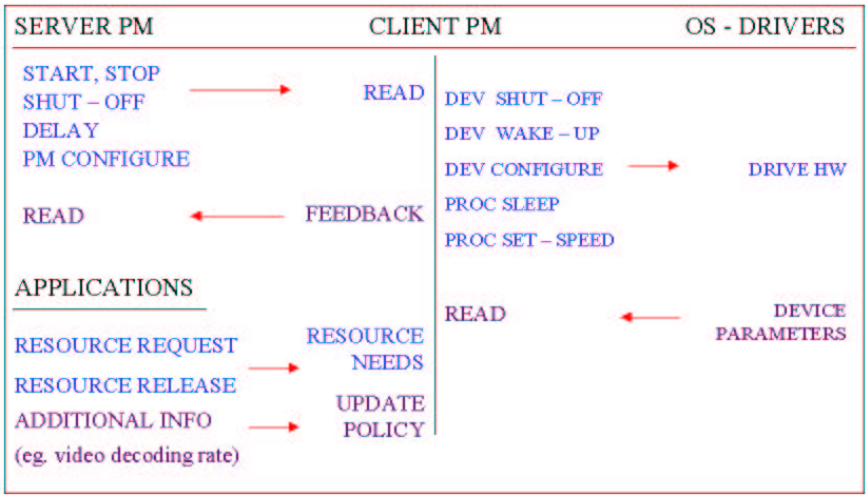


Fig. 3. Communication protocol

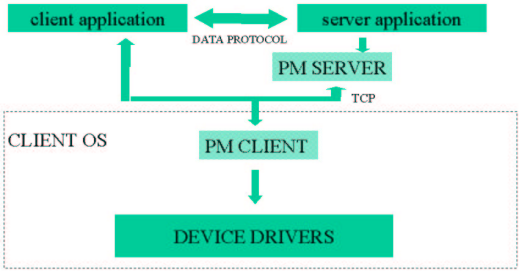


Fig. 2. Server Controlled PM Architecture

the client PM.

Figure 3 shows the communication protocol between the server and client. Control commands are issued by the server PM, interpreted by the client PM, and translated into appropriate device driver calls. Upon request from the server PM, device specific information is fetched by the client PM. Application specific information can also be retrieved by the client PM via API calls.

A. Server Power Manager

The server PM interfaces with the application in order to exploit information available on the host system, in addition to the knowledge of the overall network conditions and the specific feedback information provided by each client. Based on all this knowledge, the server PM controls the power configuration of the WLAN card by communicating with the client PM. The power control commands used include: i) switch-off the WLAN; ii) set the off time; iii) enable the 802.11b PM policy; iv) set 802.11b PM parameters, e.g. the period between wake ups and the timeout before going back to doze mode. The server PM enables:

Client Adaptation. After the application server initializes the server PM, an additional set of APIs becomes available. The initialization routine places the server PM into a state of waiting for incoming client PM requests. Each accessing client PM has a dedicated TCP connection to the server PM. The client PM informs the server PM about the application e.g. input buffer size, the expected value and variance of the service rate, as well as network interface specific information e.g. the WLAN card status and its on/off transition time.

Traffic Adaptation. Server PM monitors both the wired and wireless traffic conditions with minimum overhead. By accounting for the broadcast packet rate, the server decides when to enable the 802.11b PM. For example, in very light traffic conditions, the 802.11b PM might be used instead of a switch-off policy.

Traffic Shaping. The server PM schedules transmissions to the client in bursts, in order to compensate for the client performance and energy overheads during transitions between on and off states. The client WLAN card is switched off once the server has sent a burst of data designed to keep the application busy till the next communication burst. The burst size and delay between bursts are precomputed at the server. The delay should be large enough to almost empty the client input buffer, while the burst size should avoid overflow while keeping the buffer sufficiently filled. This maximizes the off time of the card and reduces the number of transitions between on and off states. The time for the buffer to empty, D_{burst} is the ratio of the total number of packets in the burst, $SIZE_{burst}$, to the average service rate (or buffer depletion rate) at the client, $\lambda_{s,mean}$:

$$D_{burst} = \frac{SIZE_{burst}}{\lambda_{s,mean}} \quad (1)$$

The total delay between bursts, $D^{|}_{burst}$ is:

$$D^{|}_{burst} = D_{burst} - T_{tran} - T_{cushion} \quad (2)$$

Where T_{tran} is the time needed for the transitions between WLAN on and off states, $T_{cushion}$ is the "cushion"

time so that the buffer does not ever empty completely. The cushion time helps accommodate variations of the service rate and arrival rate, 10% was found to be sufficient for most test cases. The total energy saved if the client WLAN is turned off is:

$$E_{tot} = P_{on} \cdot T_{on} + P_{off} \cdot T_{off} + P_{tran} \cdot T_{tran} \quad (3)$$

Here P_{tran} is the transition power (hardware dependent). T_{tran} can be computed by multiplying the duration of one transition to the active state, $T_{wake\text{-}up}$, by the total number of transitions, N_{tran}. The number of transitions depends on the size of the burst and the total size of streamed file.

If $\lambda_{a,mean}$ is the average arrival rate at the client input buffer, the total time to send a burst of data to the client will be:

$$T_{on} = \frac{SIZE_{burst}}{\lambda_{a,mean}} \quad (4)$$

Finally, we can obtain the total energy consumed with our methodology:

$$E_{tot} = P_{on} \cdot \frac{SIZE_{burst}}{\lambda_{a,mean}} + P_{off} \cdot \frac{SIZE_{transfer}}{\lambda_{s,mean}} + P_{tran} \cdot T_{tran} \quad (5)$$

In addition to scheduling communication so that the client WLAN can be turned off with no performance overhead, we can also perform the same scheduling while 802.11b PM is enabled. This allows us to save energy during the burst periods, but the overall burst time grows because of the decreased responsiveness and increased contention probability between data and broadcast packets.

B. Client Power Manager

The client power manager communicates with the server PM, and also interfaces with the device drivers and client applications. The main client PM tasks are:

Server Interface The client application decides when to set-up the communication between the server PM and the client PM. Once established, the client application provides the information to be forwarded to the server e.g. the buffer size and the depletion rate. The device drivers report the main characteristics of the devices to be managed, e.g. the transition time between on and off states for client WLAN.

Device Interface The client PM calls the appropriate device driver function depending on the command sent by the server PM. Possible actions taken by the client include changing the parameter of the 802.11b PM, switching the WLAN on and off, and reading the interface statistics such as the signal to noise ratio. The client PM can also interact with the CPU by changing its power mode or setting its clock speed.

Application Interface Applications can feedback information that can be exploited by both the server and the client PM. Examples include sending the current backlog level to the server, so that thee server knows exactly how much data to provide in a burst in order to refill the buffer; or providing the buffer size. The application could directly request a WLAN wake-up when its input buffer reaches a minimum value.

Application Driven Infrastructure The client PM can also be used in stand-alone mode (with no server). While in this mode, the applications provide their resource needs to the power manager. The PM then turns on or off devices appropriately. Some of the overhead needed to turn on a device can be masked, as many applications have extra latency due to the initial set-up.

C. Implementation

The implementation of the server driven power management strategy is obtained by the interaction of software modules at the user level and kernel level. At the kernel level, we implemented a power manager module (*powman*) that interfaces to the kernel and to user applications as a device driver. Applications request services by mean of the *ioctl* system calls, a standard API mechanism provided by POSIX compliant operating systems. The module can be used directly by applications running on the mobile terminal. The interaction with the server power manager is obtained by mean of an user-level daemon (*cpowmand*) which handles requests from the server application. Once started, *cpowmand* tries to establish a TCP connection to the remote server application and then it listens for the server. Each request is forwarded to the powman module using ioctls. Requests must have a format understandable by the client power manager. For this reason, we created a library of routines that can be invoked by the server that implements the communication protocol.

The power management is thus split in two components:

- A kernel level component that realizes the interaction with local applications and implements power control command by invoking peripheral driver function
- An user-level component needed to implement a server controlled policy. This organization can be eases the implementation of stand-alone policies exploiting power controls provided by our infrastructure.

It must be noted that there is not overhead due to address space separation in our case, since the interaction between user-level daemon and kernel module is limited to control messages.

In this section we described the main characteristics of server and client power managers, how they interact to provide energy efficient power management and provided some implementation details. In the next section we discuss the differences about client-centric and server-centric power management strategies.

IV. Server-Centric VS Client-Centric Approach

A. The Client-Server Paradigm

The client-server paradigm is commonly used in multimedia environments to implement audio/video streaming applications. Client application is usually multithreaded, with at least one thread used to manage remote connection and data transfer and another responsible for decoding compressed stream and playback. Server application must handle requests and send data to the clients. Depending on the multimedia format and on the application, the server may also perform some pre-processing steps on the compressed data stream before transmission, such as demultiplexing audio and video streams or extracting frame rate information. Frame rate is used by the server to schedule video packet transmission avoiding buffer overflow and ensuring continuous playback at the client. If playback rate control is performed at the client side, the server can send data at the maximum speed, but synchronization is needed to avoid client's buffer overflow. For connection-oriented transport protocol (TCP) the synchronization is automatically achieved by the zero-window mechanism, by which the server stop sending data if the client is not ready to receive them. For connection-less protocols, synchronization must be performed at the application level.

B. Power Management Strategies

Power reduction techniques tries to shut-off wireless network cards by exploiting client's buffer. When the buffer is full, the card is shut-off. The card is woken-up again when the buffer is almost empty. The wake-up instant should take into account wake-up delay to avoid a "buffer empty" condition. Key issues in these kind of policy is how to decide the wake-up instant and how to prevent the server sending data during card's off intervals. Depending on how these issues are faced, we can distinguish among two different approaches: client-centric and server-centric.

In a client-centric approach shut-down and wake-up instants can be triggered by the level of the client buffer. In [3] for example, two thresholds are established corresponding to two different buffer levels. The card receives data and accumulates in the client buffer until an high-watermark is reached. At that point, the card is shut-off while the client application consumes the backlog. When a low-watermark is reached, the card is woken-up by the client to refill the buffer. The low-watermark must take into account card's wake-up delay. However, since the consumption rate is not known (it depends on stream characteristics) threshold value should be conservative.

An advantage of this kind of policy is that in general the server does not requires modification. However, this is not always the case. In fact, when a client-centric approach is used with applications exploiting connection-less communications, the client should explicitly synchronize with the server through control messages to prevent it sending data during off intervals. As a consequence, the server must be aware of client's power conservation policy.

In a server-centric approach, the server splits data transfer into periods, each period being characterized by a data burst transmission and an idle interval during which the server does not perform data transmission. The card is shut-off after each burst and woken-up after a pre-programmed time interval. Since the server stops transmission, no explicit synchronization is needed even with connection-less communication. Moreover, the server can exploit stream-related information to enhance the efficiency of the policy. For example, in the policy we implemented, the server extract from the stream the consumption rate of data bursts (as explained in the Section V), that can be exploited to perform on-time wake-up of the wireless network card. Compared to client-centric approaches however, server-centric ones requires a more complex power management infrastructure. Moreover, client-centric approaches can easily adapt to non real-time applications, where the consumption rate does not depends only on the stream.

On the other hand, server-centric approaches are more suitable to a multi-client environment. Traffic reshaping performed by the server can be used to schedule data transmission to different clients in a power efficient way. Scheduling information at the server side can be exploited to further improve energy efficiency of the clients.

In some cases, mixed policies can be implemented. For example, traffic shaping and server-controlled shut-down can be coupled with client's controlled wake-up to handle very unpredictable consumer rate.

V. Experimental Results

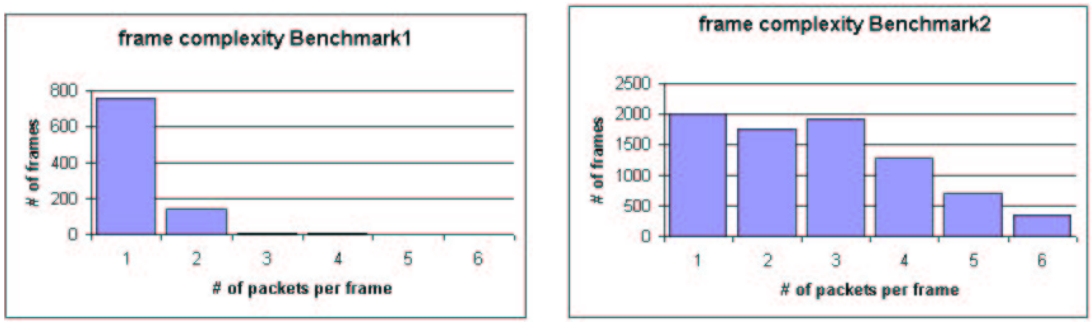
The streaming media server used for this work is a research prototype developed at Hewlett-Packard Laboratories. Real Time Streaming Protocol (RTSP) is used for session initiation and termination between the client and server. The media data units are carried using Real-time Transport Protocol (RTP) over User Datagram Protocol (UDP). Timestamps of the individual video frames are used to determine the deadline for sending data packets sent from the server to the client. The server can exploit client buffering capabilities to reschedule packet transmission times based on a cost function. In our experiments, the server transmits MPEG4 video data to a portable client device via WLAN.

We use two different benchmarks in our tests. Benchmark 1 has 12 bursts and runs at 15 frames/sec, while Benchmark 2 has 402 bursts at 30 frames/sec. The first benchmark has a large number of single packet frames, while the second benchmark has many large multi-packet frames.

The delay between bursts is:

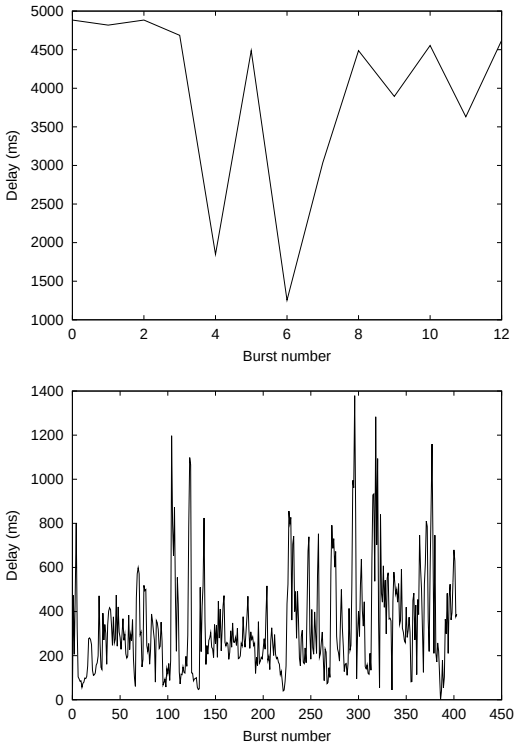
$$D_{burst} = \mathit{frame\_time} \cdot \left(\frac{n_1}{1} + \frac{n_2}{2} + \frac{n_3}{3} + \ldots + \frac{n_M}{M}\right) \quad (6)$$

Here *frame_time* is the interval between frames, n_i is the number of frames consisting of i packets and M is the maximum number of packet per frame. Figure 5 shows that the Benchmark 1 has a delay of around 4 seconds, while



a) Benchmark1 b) Benchmark2

Fig. 4. Packet histograms for each benchmark



a) Benchmark 1; b) Benchmark 2

Fig. 5. Computed burst delays for each benchmark

the Benchmark 2 has a shorter delay because the frames are more complex. Based on the delay computation, the server transmits a burst of data to the client followed by the command to switch off WLAN and the length of time until the next transmission. The client responds by turning off the WLAN and turning it back on at the pre-scheduled time. As a result, no performance penalty due to longer turn on time is incurred.

The approach was tested by measuring power consumption of a SmartBadge IV wearable device, equipped with a CISCO Aironet 350 PCMCIA WLAN card. The off/on transition time was measured at 300 ms, with an average transition power consumption of 0.1 W. The server is connected directly to an AP, and the network is completely isolated in order to perform repeatable experiments. Broadcast traffic is introduced in a controlled manner by using real traces collected from other open networks. Power measurements are performed using a DAQ board accumulating and averaging current and voltage samples (10 ksamples/sec). All measurements include the overhead imposed by our power management protocol.

First we computed the average power consumed by the WLAN card when receiving the video stream with different burst sizes for medium broadcast traffic conditions. The experiment is performed for multiple situations, WLAN with no PM, only 802.11b PM, and with server PM. From Figure V, for Benchmark 1, the server controlled approach saves 67% of average power compared to no PM, and 50% compared to the default 802.11b PM. The average power savings increase as the burst size increases, since this enables longer times between bursts, thus better compensation for the transition delay between the WLAN on and off states. In all three cases, the video plays back in real time, thus the reported average power savings directly correspond to energy savings. For Benchmark 2, the delays are very short, however, we still save 41% of power with respect to leaving the card always on and 15% with respect to the standard 802.11b power management protocol with bursts of 80 packets. We cannot use a burst size smaller than 50 packets, since the computed delays are too short and the card can

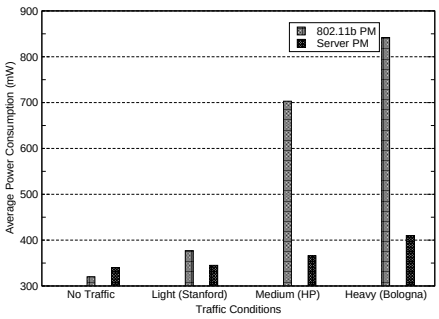
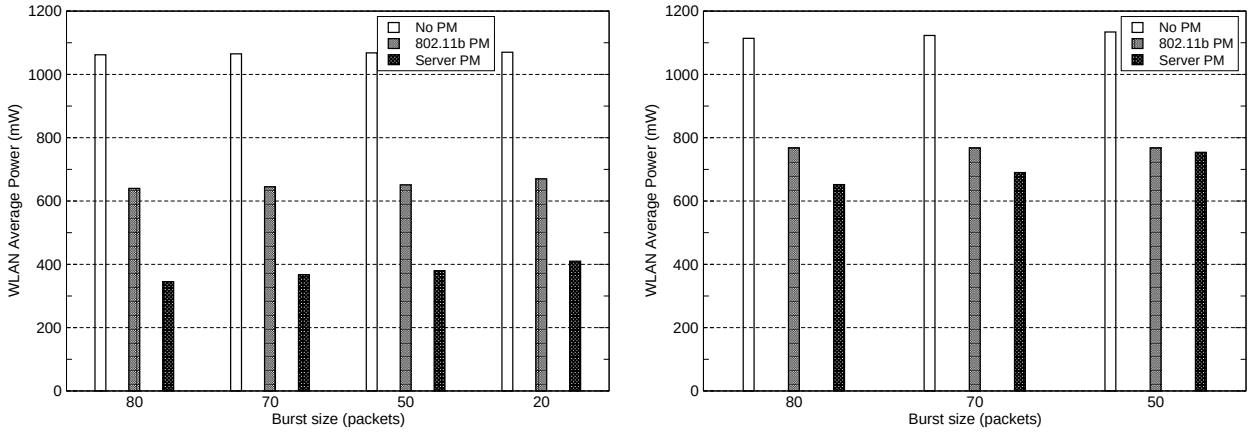


Fig. 7. Power consumption with different traffic levels

Figure 7 presents the average power consumed during streaming video under different broadcast traffic conditions. In the plot we show the difference between using only the default 802.11b PM and server controlled switch off pol-



a) Benchmark1 b) Benchmark2

Fig. 6. Power consumption for each benchmark

icy for the WLAN card (with no 802.11b PM). The switch off policy is almost insensitive to the traffic conditions, while the 802.11b PM performs better only in light traffic conditions. Clearly, our server controlled power management approach is more efficient than either policy alone as it always selects the best of the two policies.

VI. Conclusion

In this work we presented a new methodology to improve the energy efficiency of portable wireless systems that operate in a client-server fashion. Our system enables the server to exploit knowledge of the workload, traffic conditions and feedback information from the client in order to minimize client WLAN power consumption. We tested our methodology on the SmartBadge IV wearable device running a streaming video application. Our server controlled approach saves 67% of energy as compared to leaving the client WLAN card always on, and 50% as compared to the simple 802.11b PM policy.

References

[1] A. Acquaviva, L. Benini, B. Riccó, "Software Controlled Processor Speed Setting for Low-Power Streaming Multimedia," *IEEE Trans. on CAD*, Nov. 2001.

[2] F. Bellosa, "Endurix: OS-Direct Throttling of Processor Activity for Dynamic Power Management," *Technical Report TR-I4-99-03, University of Erlangen*, June 1999.

[3] D. Bertozzi, L. Benini, B. Riccó, "Power-Aware Network Interface Management for Streaming Multimedia," *Proc. of IEEE WCNC*, 2002.

[4] C. Chiasserini, P. Nuggehalli, V. Srinivasan, "Energy-Efficient Communication Protocols", *Proc. of DAC*, 2002.

[5] IEEE LAN/MAN Standards Committee, "Part 11: Wireless LAN MAC and PHY Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band", 1999.

[6] C. Jones, K. Sivalingam, P. Agrawal, J. Chen, "A Survey of Energy Efficient Network Protocols for Wireless Networks", *Proc. of DATE*, 1999.

[7] R. Krashinsky, H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown", *Proc. of MOBICOM*, 2002.

[8] R. Kravets, P. Krishnan, "Application-Driven Power Management for Mobile Communication," *Proc. of WINET*, 1998.

[9] P. Lettieri, C. Schurgers, M. Srivastava, "Adaptive Link Layer Strategies for Energy Efficient Wireless Networking", *Wireless Networks*, no. 5, 1999.

[10] J. Lorch, A. J. Smith, "Software Strategies for Portable Computer Energy Management," *IEEE Personal Communications*, June 1998.

[11] Y. Lu, L. Benini, G. De Micheli, "Operating System Directed Power Reduction," *Proc. of ISLPED*, July 2000.

[12] C. Luna, Y. Eisenberg, R. Berry, T. Pappas, A. Katsaggelos, "Transmission Energy Minimization in Wireless Video Streaming Applications," *Proc. of Asilomar Conf. on Signals, Systems, and Computers*, Nov. 2001.

[13] R. Min, A. Chandrakasan, "A Framework for Energy-Scalable Communication in High-Density Wireless Networks," *Proc. of ISLPED*, 2002.

[14] T. Pering, T. Burd, R. Brodersen, "Voltage Scheduling in the lpARM Microprocessor System", *Proc. of ISLPED*, July 2000.

[15] V. Raghunathan, S. Ganeriwal, C. Schurgers, M. Srivastava, "E^2WFQ: An Energy Efficient Fair Scheduling Policy for Wireless Systems", *Proc. of ISLPED*, 2002.

[16] J. Flinn, M. Satyanarayanan, "Energy-aware adaptation for mobile applications," *Proc. of SOSP*, Dec. 1999.

[17] P. Shenoy, P. Radkov, "Proxy-Assisted Power-Friendly Streaming to Mobile Devices," *Proc. of MMNC*, Jan. 2003.

[18] E. Shih, P. Bahl, M. Sinclair, "Dynamic Power Management for non-stationary service requests", *Proc. of MOBICOM*, 2002.

[19] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy Efficient Wireless Sensor Networks", *Proc. of SIGMOBILE*, 2001.

[20] K. Sivalingam, J. Chen, P. Agrawal, M. Srivastava, "Design and Analysis of low-power access protocols for wireless and mobile ATM networks", *Wireless Networks*, no.6, 2000.

[21] T. Simunic, L. Benini, P. Glynn, G. De Micheli, "Event-driven Power Management," *IEEE Trans. on CAD*, July 2001.

[22] M. T. Smith and G. Q. Maguire Jr., "SmartBadge/BadgePad version 4", HP Labs and Royal Institute of Technology (KTH), http://www.it.kth.se/ maguire/badge4.html, date of access: 2003-06-11.

[23] E. Takahashi, "Application Aware Scheduling for Power Management on IEEE 802.11" *Proc. of Intl. Performance, Computers, and Communications Conf.*, Feb. 2000.

[ ]Andrea Acquaviva received the Laurea degree (summa cum laude) in Electrical Engineering from the University of Ferrara, Italy, in 1999. From 1999 he is with the Department of Electronics and Computer Science (DEIS), University of Bologna as Ph.D. Student. In 2001 and 2002 he was research intern at Hewlett-Packard Laboratories (HPL), Palo Alto, CA, USA. From 2002 he is assistant professor at the Institute of Applied Information Science and Technology (ISTI), University of Urbino, Urbino, Italy. His research interests are: architectures and applications for mobile systems, embedded operating system design (with emphasis on low-power design), streaming multimedia applications (especially video/audio input/output), real-time systems, low-power design and power management of microprocessors and I/O peripherals.

[ ]Tajana Simunic Rosing Tajana Simunic Rosing is currently a full time researcher at HP Labs and is working part-time at Stanford University. At Stanford she has been involved with leading research of a number of graduate students and has taught graduate level classes. Her research interests are low-power system design, embedded systems and wireless system design. She finished her PhD in 2001 at Stanford University, concurrently with finishing her Masters in Engineering Management. Her PhD topic was Dynamic Management of Power Consumption. Prior to pursuing the PhD, she worked as a Senior Design Engineer at Altera Corporation. She obtained the MS in EE from University of Arizona. Her MS thesis topic was high-speed interconnect and driver-receiver circuit design. She has served at a number of Technical Paper Committees, and is currently an Associate Editor of IEEE Transactions on Circuits and Systems.

[ ]Sumit Roy Sumit Roy is a Senior Research Scientist at Hewlett-Packard Laboratories, Palo Alto, California. His research interests include streaming media architecture for wireless systems, content distribution and caching, and media services. He received the B.Tech. in Electrical Engineering from the Indian Institute of Technology, Bombay, India in 1991, and the M.S. and Ph.D. in Computer Engineering from Wayne State University, Detroit, Michigan, in 1996 and 1998, respectively. He was a Research Assistant Professor at the Department of Electrical and Computer Engineering, Wayne State University from 1998 - 1999. ¿From 1991 - 1993 he worked as a Field Engineer with Halliburton Logging Services, Inc.

[ ]Vinay Deolalikar Vinay Deolalikar obtained his Ph.D. in electrical engineering from the university of southern california in 1999. From 1999 to 2000, he was with Hughes Network Systems in San Diego. Since August 2000, he has been with the information theory group at HP Labs, Palo Alto. His research interests are in communications, security, and computational complexity and the applications of number theory and algebraic geometry in these disciplines.